CVAR SZÁMÍTÁS SRA ALGORITMUSSAL[1]

ÁGOSTON KOLOS CSABA
Budapesti Corvinus Egyetem

citation and similar papers at core.ac.uk

A $CVaR$ kockázati mérték egyre nagyobb jelentőségre tesz szert portfóliók kockázatának megítélésekor. A portfolió egészére a $CVaR$ kockázati mérték minimalizálását meg lehet fogalmazni kétlépcsős sztochasztikus feladatként. Az SRA algoritmus egy mostanában kifejlesztett megoldó algoritmus sztochasztikus programozási feladatok optimalizálására. Ebben a cikkben az SRA algoritmussal oldottam meg $CVaR$ kockázati mérték minimalizálást[2].

1 Bevezetés

Egy értékpapír vagy portfólió kockázatának mérése régóta foglalkoztatja az elméleti és gyakorlati pénzügyi szakembereket. A kockázat mérésére több megoldás is létezett. Ezek közül az ún. VaR (Value-at-Risk, magyarul kockáztatott érték) kockázatmérték terjedt el leginkább. A VaR_β kockázatmérték megadja azt az értéket, amelynél a döntéshozó β valószínűséggel nem veszít többet. A VaR népszerűségének oka a könnyű értelmezhetőség, bár számos elméleti és numerikus probléma merült fel. Elméleti oldalról probléma, hogy a VaR ugyan megadja azt az értéket, amelynél a döntéshozó β valószínűséggel nem veszít többet, de arról nem mond semmit, hogy a veszteség mennyivel haladja meg ezt az értéket, ha a veszteség mégis nagyobb ennél az értéknél. Szintén elméleti probléma, hogy diverzifikációval akár nőhet is a VaR, tehát nem teljesül a kockázatmértéktől elvárt szubadditivitás követelménye (lásd: [2]). Numerikus oldalról probléma, hogy optimalizálás esetén a VaR modellek nemkonvex optimalizálási feladatokra vezethetnek, amelyek köztudottan nem jól kezelhetők.

Érdemes megemlíteni, hogy egy VaR-korlát ($P\{Y \geq K\} \geq p$) tulajdonképpen valószínűségi korlát, ami gyakran használt megoldás a sztochasztikus programozásban. Ezekkel kapcsolatban Prékopa András végzett kiterjedt kutatásokat, nevezetesen megmutatta, hogy egy sor valószínűségi eloszlás esetén az eloszlás sűrűségfüggvénye logkonkáv és ezért az eloszlásfüggvénye is logkonkáv. Ezért a valószínűségi korlát által megadott megengedett megoldások halmaza ($\{\mathbf{x}|P\{\mathbf{Y} \geq \mathbf{x}\} \geq p\}$) konvex, ha a korlát megfogalmazásában konvex függvények szerepelnek (lásd pl.: [11]). Logkonkáv eloszlás például a nem degenerált normális eloszlás, a Dirichlet eloszlás és a Wishart eloszlás is. Sajnos az általam vizsgált portfólió választási feladatok nem tartoznak az

[1]Beérkezett: 2010. január 18. E-mail: kolos.agoston@uni-corvinus.hu.
[2]Ezúton szeretnék köszönetet mondani Deák Istvánnak segítségéért. Szeretném továbbá megköszönni két ismeretlen lektorom hasznos tanácsait is.

említett feladat-osztályba, mert a döntési változók és a véletlen paraméterek szorzatát kell számolni.

A $CVaR$ (Conditional VaR, magyarul feltételes kockáztatott érték) kezeli ezeket a problémákat. A $CVaR$ matematikailag egy feltételes várható érték, tehát figyelembe veszi a veszteség nagyságát is, és a $CVaR$ koherens kockázatmérték (lásd: [10,2]).

A $CVaR$ modelleknek egyik fontos iránya az ún. portfólióoptimalizálási modellek. Ezen modellek esetében a döntéshozó a portfólió kockázatát ($CVaR$) szeretné minimalizálni bizonyos feltételek mellett. Rockafellar és Uryasev ([12]) hozzájárulása jelentős a területhez, akik a $CVaR$ kockázati mérték minimalizálást lineáris programozási feladatként fogalmazták meg. Künzi-Bay és Mayer ([8]) a $CVaR$ kockázati mérték minimalizálását kétlépcsős sztochasztikus feladatként írta fel. Az ő felírásukkal a kétlépcsős sztochasztikus modellek megoldására alkalmas algoritmusokkal is meg lehet oldani a modellt. Az eljárást továbbfejlesztette Fábián, aki többperiódusú portfólió modelleket oldott meg ([7]).

Sztochasztikus programozási feladatok megoldására (nemcsak kétlépcsős, hanem egyéb típusúakra is) egy új heurisztikus algoritmus a Deák által kifejlesztett SRA algoritmus ([3,6,4,5]), amely alkalmas akár nagyméretű sztochasztikus feladatok megoldására is. Ebben a cikkben az SRA algoritmust használtam $CVaR$ kockázati mérték minimalizálására.

A cikk felépítése a következő: a 2. fejezetben a $CVaR$ kockázati mérték minimalizálására felírt portfólió modellt mutatom be, a 3. fejezetben az SRA algoritmust ismertetem röviden. A 4. fejezetben az SRA algoritmus implementálását mutatom be $CVaR$ kockázati mérték minimalizálására. A számítási eredményeket az 5. fejezet mutatja be.

A cikkben a következő jelöléseket alkalmazom: $\mathbf{x}$ vektor i-edik koordinátáját x_i jelöli. Y valószínűségi változót, $\mathbf{Y}$ pedig valószínűségi vektorváltozót jelöl. $\tilde{\mathbf{Y}}^j$ az $\mathbf{Y}$ valószínűségi vektorváltozó egy realizációját jelöli, ennek i-edik koordinátája pedig $\tilde{Y}_i^j$.

2 A CVaR modell

Legyen Y egy valószínűségi változó, amely egy döntéshozó lehetséges (pénzben mért) veszteségét (a nyereség negatív veszteségként értelmezendő) fejezi ki. Az egyszerűség kedvéért legyen Y folytonos valószínűségi változó, melynek sűrűségfüggvénye $f(y)$, eloszlásfüggvénye pedig $F(y)$. Ehhez az Y valószínűségi változóhoz tartozó VaR_β kockázati mérték megadja azt az értéket, amelynél a döntéshozó β valószínűséggel nem veszít többet:

$$VaR_\beta = F^{-1}(1-\beta),$$

ahol $F^{-1}(.)$ az $F(y)$ eloszlásfüggvény általánosított inverze:

$$F^{-1}(w) = \inf_y \big\{F(y) \geq w\big\}$$

A döntéshozó vagyonát jellemzően nem egy értékpapírba helyezi, ezért Y több valószínűségi változó összegeként áll elő. Portfólió optimalizálási feladatok esetén a döntéshozó n értékpapírba fektetheti tőkéjét, ezek jövőbeni értékét jelölje Y_i valószínűségi változó. A döntéshozó $x_1, x_2, ...x_n$ összegeket fektet az értékpapírokba. A jövőbeli vagyont ekkor az $\sum_{i=1}^{n} x_i Y_i$ összeg adja meg, tehát $Y = -\sum_{i=1}^{n} x_i Y_i$. Az egyszerűség kedvéért tegyük fel, hogy a döntéshozó egységnyi tőkével rendelkezik, ekkor x_i változók az optimális választás esetén az eszközök arányát mutatják, Y_i valószínűségi változók pedig az eszköz hozamát.

Ugyanehhez az Y valószínűségi változóhoz tartozó $CVaR_\beta$ feltételes kockázati mérték megadja, hogy várhatóan mennyit veszít a döntéshozó, ha a veszteség meghaladja VaR_β értéket:

$$CVaR_\beta = E(Y|Y \geq VaR_\beta) = E(Y|Y \geq F^{-1}(\beta)) \tag{1}$$

Rockafellar és Uryasev ([12]) megmutatta, hogy folytonos valószínűségi változók esetén a $CVaR_\beta$ a

$$\min_z z + (1-\beta)^{-1} E([Y-z]^+) \tag{2}$$

feladat megoldásaként is megkapható, ahol $[x]^+$ x pozitív részét jelöli. Megmutatható, hogy a (2) kifejezés optimumhelye[3] (z változó optimális értéke) VaR_β.

Amennyiben Y nem folytonos valószínűségi változó az (1) képlettel megadott definíció további pontosításra szorul[4], és ráadásul a (2) minimuma nem feltétlenül egyezik meg az (1) képlettel megadott értékkel, ezért Pflug [10] azt javasolja, hogy a (2) képletet tekintsük definíciónak. Pflug olyan eloszlásokkal foglalkozik, ahol az eloszlásfüggvényben ugrások lehetnek. Általános eloszlásokra Rockafellar és Uryasev [13] dolgozta ki az elméletet.

Portfólió optimalizálás esetén szeretnénk minimalizálni $CVaR_\beta$ kockázati mértéket, feltéve hogy a döntéshozónak van valamekkora hozamelvárása (r^*). Most ezt a feladatot kétlépcsős modell segítségével írjuk fel[5]. Ekkor a döntési probléma:

$$\min_{\mathbf{X},z} \mathbf{c}^T\mathbf{x} + z + E(Q_C(\mathbf{x}, z, \mathbf{Y})),$$

feltéve, hogy:

$$\sum_{i=1}^{n} x_i = 1,$$

$$\sum_{i=1}^{n} x_i E(Y_i) \geq r^*.$$

A második lépcső:

$$Q_C(\mathbf{x}, z, \mathbf{Y}) = (1-\beta)^{-1} \min_y y,$$

[3]Elképzelhető, hogy az optimumhely nem egyértelmű, a részletekről lásd.: [13]

[4]A részletekről lásd: [13]

[5]A felírást Künzi-Bay és Mayer ([8]) adta meg.

feltéve, hogy:

$$y \geq -\sum_{i=1}^{n} x_i Y_i - z,$$

$$y \geq 0,$$

ahol Y_i valószínűségi változó az i-edik eszköz hozamát mutatja, x_i döntési változó az i-edik eszközbe fektetett tőke arányát jelenti, z az optimalizáláshoz használt segédváltozó, β pedig a $CVaR$ kockázati mértékhez tartozó külső paraméter (megbízhatósági szint). Az x_i döntési változókra feltehetünk nemnegativitási korlátot, de ez technikailag nem szükséges (meg lehet engedni fedezetlen eladásokat is).

Amennyiben $Y_1, Y_2, ..., Y_n$ valószínűségi változók diszkrét eloszlásúak (és korlátosak), akkor a kétlépcsős feladatot meg lehet oldani lineáris programozási feladatként (lásd: [12]). Elterjedt módszer, hogy folytonos valószínűségi változókat is diszkretizálnak (vagy mintát vesznek), és így lineáris programozási feladatként oldják meg. Kihasználva a specialitásokat Künzi-Bay és Mayer ([8]) megadott egy hatékony eljárást a $CVaR$ optimalizálások esetére. Ugyanakkor a diszkretizálás magas dimenziók (sok eszköz) esetén problematikus lehet (lásd: [4]).

Másik lehetséges módszer kétlépcsős sztochasztikus problémák megoldására a Monte Carlo integrálásos technikák, amelynek egyik képviselője az SRA algoritmus.

3 Az SRA algoritmus

Az SRA (Successive Regression Approximations) egy mostanában kifejlesztett heurisztikus algoritmus sztochasztikus programozási feladatok megoldására[6]. Ezek közül most csak a kétlépcsős programozási feladatok megoldását mutatom be.

Tekintsünk egy kétlépcsős sztochasztikus programozási feladatot az alábbi formában:

$$\min_{\mathbf{x}} \mathbf{c}^T \mathbf{x} + E(Q_C(\mathbf{x}, \mathbf{Z})),$$

feltéve, hogy:

$$A\mathbf{x} = \mathbf{b},$$

$$\mathbf{x} \geq \mathbf{0}.$$

A második lépcső:

$$Q_C(\mathbf{x}, \mathbf{Z}) = \min_{\mathbf{y}} \mathbf{q}^T \mathbf{y},$$

feltéve, hogy:

$$A\mathbf{x} + W\mathbf{y} = \mathbf{Z},$$

$$\mathbf{y} \geq 0.$$

[6] Az algoritmust Deák István fejlesztette ki.

A feladatban a nehézséget a $E(Q_C(\mathbf{x}, \mathbf{Z}))$ várható érték kiszámítása jelenti. A várható érték kiszámítása problematikus, de egy konkrét pontban a függvényre nem nehéz torzítatlan becslést adni: legyen $\tilde{\mathbf{Z}}^1$, $\tilde{\mathbf{Z}}^2$, ..., $\tilde{\mathbf{Z}}^k$ a $\mathbf{Z}$ valószínűségi vektorváltozó k független realizációja, ekkor

$$p(\mathbf{x}) = \frac{1}{k} \sum_{i=1}^{k} Q_C(\mathbf{x}, \tilde{\mathbf{Z}}^i). \tag{3}$$

az $E(Q_C(\mathbf{x}, \mathbf{Z}))$ értéknek egy torzítatlan becslése[7].

Az SRA algoritmus alapgondolata az, hogy az $E(Q_C(\mathbf{x}, \mathbf{Z}))$ nehezen kiszámítható függvényt egy kvadratikus függvénnyel közelíti, és az optimum közelében elkezdi 'pontosítani' ezt a függvényt. Az algoritmus indulásához szükségünk van kezdőpontokra. Véletlenszerűen felveszünk $\mathbf{x}^i$ kezdőpontokat (mondjuk l darabot), és ezekre a kezdőpontokra kiszámítjuk a $p_i(\mathbf{x}^i)$ becsléseket. Rendelkezünk $S_l = \{\mathbf{x}^i, p_i(\mathbf{x}^i)\}_{i=0}^{l-1}$ pontok halmazával, ezekre a pontokra egy

$$q_l(\mathbf{x}) = \mathbf{x}^T D_l \mathbf{x} + \mathbf{b}_l^T \mathbf{x} + c_l.$$

alakú kvadratikus függvényt illesztünk.

Az eredeti első lépcső feladatot helyettesítjük a

$$\min_{\mathbf{x}} \mathbf{c}^{\mathbf{T}} \mathbf{x} + q_l(\mathbf{x}),$$

feltéve, hogy:

$$A\mathbf{x} = \mathbf{b},$$
$$\mathbf{x} \geq 0$$

feladattal, ami kvadratikus programozási feladat. Kiszámítjuk a feladat optimális megoldását. Ha a kapott pont 'elég jó', megállunk, ha nem, kiszámítjuk az optimumhoz a $p(\mathbf{x})$ becslést, hozzávesszük az eddigi pontokhoz, és visszatérünk a kvadratikus közelítéshez (az algoritmus részletesebb leírása megtalálható: [6,4,5]).

Az 'elég jó' megállási kritérium lehet valamilyen pontosság (statisztikai hibahatár) megkövetelése (lásd például: [9]).

Az SRA algoritmus jól teljesít sztochasztikus problémák megoldásában, de hiányzik az elméleti bizonyítása. A cikk következő részében azt szándékozom demonstrálni, hogy $CVaR$ kockázati mérték minimalizálására is használható az algoritmus.

4 Az SRA algoritmus implementálása

4.1 Alapfeladat

A $CVaR$ kockázati mérték minimalizálási feladatot a következő formában szokás felírni:

$$\min_{\mathbf{x}, z} z + E(Q_C(\mathbf{x}, z, \mathbf{Y})),$$

[7] A gyakorlatban nem ezt a becslés célszerű alkalmazni. A részletekről lásd: [6]

feltéve, hogy:

$$\sum_{i=1}^{n} x_i = 1,$$

$$\sum_{i=1}^{n} x_i E(Y_i) \geq r^*.$$

A második lépcső:

$$Q_C(\mathbf{x}, z, \mathbf{Y}) = (1-\beta)^{-1} \min_y y,$$

feltéve, hogy:

$$y \geq -\sum_{i=1}^{n} x_i Y_i - z,$$

$$y \geq 0,$$

ahol Y_i valószínűségi változó az i-edik eszköz hozamát mutatja, x_i döntési változó az i-edik eszközbe fektetett tőke arányát jelenti, z az optimalizáláshoz használt segédváltozó, β pedig a $CVaR$ kockázati mértékhez tartozó külső paraméter (megbízhatósági szint).

Természetesen a konkrét optimalizáláshoz szükségünk van $\tilde{\mathbf{Y}}^1$, $\tilde{\mathbf{Y}}^2$, ..., $\tilde{\mathbf{Y}}^q$ realizációkra. A $Q_C(\mathbf{x}, z, \mathbf{Y})$ függvényt a mintaátlaggal helyettesítjük, ekkor a második lépcső a következő alakot ölti:

$$Q_C(\mathbf{x}, z, \tilde{\mathbf{Y}}^1, \tilde{\mathbf{Y}}^2, ..., \tilde{\mathbf{Y}}^q) = \frac{1}{(1-\beta)q} \min_{\mathbf{y}} \sum_{j=1}^{q} y_j,$$

feltéve, hogy:

$$y_j + z \geq -\sum_{i=1}^{n} x_i \tilde{Y}_i^j, \qquad j = 1...q,$$

$$y_j \geq 0, \qquad j = 1...q$$

Ezen a felíráson csak annyiban változtattam, hogy a z változóban végzett optimalizálást nem az első, hanem a második lépcsőben végeztem. Az általam megoldott feladat[8][9]:

$$\min_{\mathbf{X}} E\big(Q_C(\mathbf{x}, \tilde{\mathbf{Y}}^1, \tilde{\mathbf{Y}}^2, ..., \tilde{\mathbf{Y}}^q)\big),$$

feltéve, hogy:

$$\sum_{i=1}^{n} x_i = 1,$$

[8] A célfüggvényhez hozzá lehet adni egy lineáris költségtagot, továbbá az első lépcsőhöz tetszőleges lineáris korlát is hozzáírható, az algoritmus változatlan formában működik.

[9] A célfüggvény helyett az SRA algoritmus implementálásakor itt is k minta átlaga áll. A részletek a 4.5. alfejezetben vannak kifejtve, a célfüggvény pontos meghatározása a (4) képletben található.

$$\sum_{i=1}^{n} x_i E(Y_i) \geq r^*.$$

A második lépcső:

$$Q_C(\mathbf{x}, \tilde{\mathbf{Y}}^1, \tilde{\mathbf{Y}}^2, ..., \tilde{\mathbf{Y}}^q) = \min_{z,\mathbf{y}} z + \frac{1}{(1-\beta)q} \sum_{j=1}^{q} y_j,$$

feltéve, hogy:

$$y_j + z \geq -\sum_{i=1}^{n} x_i \tilde{Y}_i^j, \qquad j = 1...q,$$

$$z, y_j \geq 0, \qquad j = 1...q.$$

A változtatásnak az az értelme, hogy a második lépcső így egy $CVaR$ számítás, a portfólió optimalizálás pedig az első lépcsőben történik. A második lépcső kiszámítását nem kell lineáris programozási feladatként megoldani. A második lépcsőhöz tartozó lineáris programozási feladat optimális megoldása megadja a portfólió veszteségének realizációi közül azok arányát, amelyek esetén a veszteségek a felső β kvantilisbe esnek, ami viszont lineáris programozási feladat nélkül is kiszámítható, jelentős futási időt spórolva.

Ennek a kétlépcsős sztochasztikus feladatnak a megoldására a 3. fejezetben leírt SRA algoritmust használtam, kisebb változtatásokkal.

4.2 Kezdő pontok meghatározása

Az első változtatás a kezdő pontok megválasztásánál történt: olyan induló pontokat választottam, amelyek rajta vannak az első lépcső korlátai által kifeszített hipersíkon. További változtatás, hogy kijelöltem egy középpontot, és a véletlenül felvett pontok e középpont körül helyezkednek el. A középpont az első lépcső korlátai által kifeszített hipersík origóhoz legközelebbi pontja[10]. Az volt mögötte a heurisztikus megfontolás, hogy a diverzifikálás előnyei miatt – nem szélsőséges esetben – az optimum is valahol az egységszimplex 'közepén' lesz, így a megfelelő középpont környékén felvett pontok jól leírják a pótlás feladat közelítését.

4.3 Kvadratikus közelítés

Az eredeti SRA algoritmus fontos jellemzője, hogy a regressziós közelítés előállításánál minden korábbi pontot felhasznál, így a közelítés egyre pontosabbá válik. Mégis fontos, hogy az optimum közelében lévő pontokat jobban figyelembe vegyük, mint az optimumtól távolabb lévő pontokat. Az eredeti SRA algoritmus ezt a követelményt úgy oldja meg, hogy minden ponthoz egy súlyt rendel, annak függvényében, hogy (vélhetően) mennyire van távol

[10] A korlátok között mindig szerepel a $\sum_{i=1}^{n} x_i = 1$ feltétel, tehát ez a pont rajta van az egységszimplexen.

az optimumtól. Az implementálásnál más utat követtem: amennyiben elegendő pont áll rendelkezésre, az induló pontokat (de csak azokat) kihagytam a kvadratikus közelítés előállításánál. A heurisztikus gondolat az volt ezen eljárás mögött, hogy induláskor szükség van a generált pontok szóródására, hogy a négyzetes közelítés felvegye a konvex kvadratikus alakot. Amikor viszont az algoritmus már 'kitapogatta' az optimum körülbeli helyét, az optimumtól távol lévő pontok már csak hátráltatnak.

4.4 A CVaR becslés torzítottsága

Felmerült az a probléma, hogy a pótlás feladat optimális célfüggvénye torzítottan becsüli $CVaR_\beta$ értéket. Az *1. táblázat* számszerűen szemlélteti a torzítás mértékét sztenderd normális eloszlás esetén. A táblázatban az elemszám azt mutatja, hogy hány generált véletlen szám alapján számoltam a $CVaR_{0,9}$ becslését. A becslési eljárást megismételtem 10000-szer minden elemszám esetén. A táblázat második oszlopa mutatja $CVaR_{0,9}$ becslések átlagát, a harmadik a 95%-os konfidencia intervallumot, a negyedik pedig egy becsléshez szükséges idő átlagát (másodpercben). A táblázatból jól látszik, hogy az elemszám növekedésével csökken a torzítás mértéke, de lineárisnál gyorsabban nő a szükséges idő.

			95%-os konfidencia intervallum		
Elemszám	Elméleti érték	Átlag	Alsó határa	Felső határa	Idő
100	1,755	1,734	1,730	1,738	0,00004
500	1,755	1,750	1,749	1,752	0,00022
2500	1,755	1,754	1,753	1,755	0,00206
12500	1,755	1,755	1,754	1,755	0,03370
62500	1,755	1,755	1,754	1,755	0,70220

1. táblázat. A $CVaR$ becslés torzítása. Az első oszlop megadja a becsléshez generált véletlenszámok számát, a második oszlopban szerepel az elméleti érték, a harmadikban a $CVaR$ becslések átlaga, a negyedik és ötödikben a 95%-os konfidencia intervallum alsó és felső határa, a hatodik oszlopban pedig a becsléshez szükséges idő átlaga szerepel másodpercben.

Fontos hangsúlyozni, hogy a torzítás nem az SRA algoritmus következménye, az akkor is jelen van, ha a $CVaR$ kockázati mérték optimalizálási feladatot lineáris programozási feladatként oldjuk meg[11].

4.5 Az $Q_C(\mathbf{x}, \mathbf{Y})$ érték becslése

A 3. fejezetben a (3) képlettel adtunk egy becslést a második lépcső célfüggvényének értékére. Jelen $CVaR$ kockázati mérték optimalizálás esetén a második lépcső becslése úgy történik, hogy generálunk véletlen számokat és vesszük a felső β kvantilis átlagát. Kérdés, hogy hány számot generáljunk, megismételjük-e az eljárást, és ha igen, hányszor. Egy $CVaR$ becsléshez

[11] Megjegyezzük, hogy az 1. táblázat eredményei összhangban vannak Mak Morton és Wood [9] elméleti eredményeivel.

generált változók számát elemszámnak hívom a továbbiakban. Az elemszámot úgy kell megválasztani, hogy kellően nagy legyen a torzítás megfelelő csökkentése céljából, ugyanakkor ne legyen a szükségesnél nagyobb, a futási idő miatt. Egy $CVaR$ becslés ingadozása még akkor is jelentős, ha a torzítás már elenyésző (lásd 1. táblázat). Emiatt célszerű még viszonylag nagy elemszám esetén is több $CVaR$ becslésnek venni az átlagát. Ismétlések számaként fogok arra utalni, hogy pontosan hány $CVaR$ becslésnek veszem az átlagát. Képletben:

$$p(\mathbf{x}) = \frac{1}{k}\sum_{i=1}^{k} Q_C(\mathbf{x}, \tilde{\mathbf{Y}}^{1,i}, \tilde{\mathbf{Y}}^{2,i}, ..., \tilde{\mathbf{Y}}^{q,i}), \tag{4}$$

ahol q az elemszám, k pedig az ismétlések száma.

Az elemszám és ismétlések száma kívülről adott paraméter, amelyet a döntéshozó által elvárt pontosság alapján lehet meghatározni.

5 Számítási eredmények

A kutatás jelen szakaszában a Rockafellar és Uryasev ([12]) cikkben közölt adatokkal számoltam, hogy az eredményeket lehessen más eredményekhez viszonyítani.

Rockafellar és Uryasev ([12]) a cikkükben 3 eszközt vizsgálnak: S&P 500 részvényindex (S&P 500), hosszú távú amerikai államkötvény portfólió (Gov Bond) és kis tőkésítettségű amerikai vállalati portfólió (Small Cap). Az eszközök esetén a várható értéket és a szórást a *2. és 3. tábla* mutatja.

Eszköz	Átlagos hozam
S&P 500	0,0101110
Gov Bond	0,0043532
Small Cap	0,0137058

2. táblázat. Az eszközök hozama

	S&P 500	Gov Bond	Small Cap
S&P 500	0,00324625	0,00022983	0,00420395
Gov Bond	0,00022983	0,00049937	0,00019247
Small Cap	0,00420395	0,00019247	0,00764097

3. táblázat. A portfólió kovarianciamátrixa

A 3 eszköz eloszlására együttes normális eloszlást tételeznek fel. Együttes normális eloszlás esetén a $CVaR$ optimális portfólió és a Markowitz optimális portfólió egybeesik (lásd: [12]). A Markowitz-féle modell megoldása egy kvadratikus optimalizálás eredménye, aminek az értékét a *4. táblázat* mutatja. Az optimális portfólió esetén a 90%-os, 95%-os és 99%-os $CVaR$ értékeket az *5. táblázat* adja meg.

S&P 500	Gov Bond	Small Cap
0,452013	0,115573	0,432414

4. táblázat. Optimális eszközsúlyok

$\beta = 0,9$	$\beta = 0,95$	$\beta = 0,99$
0,096975	0,115908	0,152977

5. táblázat. $CVaR$ értékek az optimális portfólióra

Rockafellar es Uryasev ([12]) közöl futási eredményeket, de minden beállításról csak egyet. A *6. táblázatban* olyan eredményeket közlök, amelyek minden elemszámhoz 100 futás eredményeit összegzik, így az optimum minőségéről jobb képet kapunk. Rockafellar es Uryasev CPLEX solvert használt, én viszont MINOS megoldót. Az eredményeket azért is közlöm, mert így a különbségek az algoritmusok különbségének tudhatók be és nem a solverek különbségének (és nem mellesleg ugyanazon a gépen futtattam mindkét alternatívát). A 6. táblázat különböző elemszámok esetén mutatja az algoritmus futási eredményeit. Az első oszlop az elemszámot mutatja, a második a $CVaR_{0,9}$ becslések átlaga. Látható, hogy ha az elemszám kicsi, a $CVaR$ becslés lefelé torzít. Mivel a feladat 3 eszközt tartalmaz és két korlátot, ezért az eszközök közül csak egynek az értékét (arányát) lehet szabadon megválasztani (S&P 500), ennek átlagát mutatja a 6. táblázatban a harmadik oszlop. A negyedik oszlop a futáshoz szükséges idő átlaga másodpercben. Az átlagértékek alatt zárójelben a szórások szerepelnek.

Elemszám	$CVaR$	S&P 500	Idő
100	0,09251	0,38099	0,0
	(0,01169)	(0,26894)	(0,0)
500	0,09676	0,43688	0,0
	(0,00557)	(0,15367)	(0,0)
2500	0,09725	0,45195	1,4
	(0,00234)	(0,07267)	(0,1)
12500	0,09702	0,45557	58,9
	(0,00095)	(0,03232)	(6,3)

6. táblázat. Futási eredmények – lineáris programozási feladat. Az első oszlop az optimalizáláshoz használt minta elemszámát mutatja, a második a $CVaR$ becslések átlagát, a harmadik az optimalizálás során az S&P 500 eszköz optimális értékeinek átlagát, a negyedik oszlop pedig az optimalizáláshoz szükséges idő átlagát másodpercben. Az átlagértékek alatt zárójelben a szórás szerepel.

Az SRA algoritmushoz a kódot Lahey Fortran nyelvben írtam meg. A algoritmushoz szükséges solver a MINOS. A futtatásokat egy 1,6 GHz AMD Sempron számítógépen végeztem.

A *7. táblázat* különböző beállítások esetén mutatja meg az algoritmus futási eredményeit. Az első oszlop az elemszámot mutatja, a második pedig azt, hogy egy $p_i(\mathbf{x}^i)$ érték előállításához hány becslésnek vettem az átlagát (ismétlések száma). A harmadik oszlopban szerepelnek a $CVaR_{0,9}$ becslések

átlagai. Látható, hogy ha az elemszám kicsi, a $CVaR$ becslés itt is lefelé torzít. Mivel a feladat 3 eszközt tartalmaz és két korlátot, ezért az eszközök közül csak egynek az értékét (arányát) lehet szabadon megválasztani, az algoritmus az első eszközt választja meg. Ennek átlagát mutatja a 7. táblázatban a negyedik oszlop. Az ötödik oszlop azt mutatja, hogy hány iteráció után áll le az algoritmus, a hatodik pedig a futáshoz szükséges idő másodpercben. Az átlagértékek alatt zárójelben a szórások szerepelnek.

Elemszám	Ismétlés	$CVaR$	S&P 500	# Iteráció	Idő
100	100	0,09572	0,45479	4614	16,6
		(0,00113)	(0,01620)	(1564)	(5,1)
100	1000	0,09536	0,45234	1924	60,4
		(0,00036)	(0,00828)	(608)	(20,9)
100	10000	0,09542	0,45230	829	244,3
		(0,00012)	(0,00398)	(251)	(73,9)
1000	100	0,09678	0,45314	1988	127,5
		(0,00034)	(0,00830)	(610)	(39,1)
1000	1000	0,09683	0,45178	876	554,8
		(0,00012)	(0,00389)	(211)	(134,1)
1000	10000	0,09682	0,45216	346	2189,1
		(0,00004)	(0,00200)	(140)	(884,7)
10000	10	0,09691	0,45388	1914	858,1
		(0,00039)	(0,01036)	(588)	(337,0)

7. táblázat. Futási eredmények – SRA algoritmus. Az első és második oszlop az optimalizáláshoz használt minta elemszámát és az ismétlések számát mutatja, a harmadik a $CVaR$ becslések átlagát, a negyedik az optimalizálás során az S&P 500 eszköz optimális értékeinek átlagát, az ötödik a szükséges iterációk számának átlagát, a hatodik oszlop pedig az optimalizáláshoz szükséges idő átlagát másodpercben. Az átlagértékek alatt zárójelben a szórás szerepel.

A 7. táblázatból jól látszik, hogy az SRA algoritmus képes az optimalizálási feladat megoldására. A táblázatból jól látszik, hogy ha növeljük az ismétlések számát, vagy az elemszámot, akkor pontosabb eredményt kapunk. Jól látszik az a kettőség is, hogy ha az a célunk, hogy a $CVaR$ értéket pontosan megkapjuk, akkor az elemszámot kell növelni, ha a viszont az optimális portfólió megtalálása a célunk, akkor kisebb elemszámot és több ismétlést kell választani.

Érdemes a futási eredményeket a lineáris programozási algoritmus futási eredményeivel összehasonlítani. Az összehasonlításnál nem az azonos elemszámot kell összehasonlítani, hiszen más a tartalma az elemszámnak a két esetben. Sokkal szerencsésebb, ha úgy hasonlítjuk össze az eredményeket, hogy azonos futási idő alatt milyen pontosságot ér el az algoritmus. Például lineáris programozási feladat esetén 12500-as elemszám nagyjából ugyanannyi időt igényel, mint az SRA algoritmus 100 elemszámmal és 1000 ismétléssel. A vizsgált esetben a lineáris programozási feladat kisebb torzítással (0,09702 vs. 0,09536), de nagyobb szórással (0,00095 vs. 0,00036) becsüli a $CVaR$ értéket. Az optimális portfólió megtalálásánál egyértelműen az SRA algoritmus a jobb. A lineáris programozási feladat esetében az optimális portfóliósúlyokat csak nagy szórással tudja meghatározni az algoritmus. Nagyobb elemszám választása viszont lényegesen meghosszabbítja a futási időt.

A közölt futási eredményekből messzemenő következtetéseket nem érdemes levonni, de annyit ki lehet jelenteni, hogy az SRA algoritmus versenyképes a Rockafellar es Uryasev ([12]) által felírt lineáris programozási feladattal.

6 Összefoglalás

Ebben a cikkben $CVaR$ portfólió optimalizálási feladatot oldottam meg SRA algoritmussal. Numerikus futtatási adatok alapján kijelenthető, hogy az algoritmus képes elvégezni az optimalizálási feladatot és az is, hogy az algoritmus versenyképes a lineáris programozási feladatként való felírással.

Lehetséges továbblépési irány annak felhasználása, hogy az SRA algoritmus nem csak kétlépcsős sztochasztikus feladatok megoldására képes, hanem pl. valószínűséggel korlátozott feladatok megoldására is. Ez megnyitja az utat afelé, hogy az adatokban meglévő bizonytalanságot figyelembe vegyük az optimalizációnál.

Irodalom

1. F. Andersson, H. Mausser, D. Rosen, S. Uryasev (2001): Credit risk optimization with Conditional Value-at-Risk criterion. *Mathematical Programming*, Series B, 89, 273–291.
2. P. Artzner, F. Delbaen, J.-M. Eber, D. Heath (1998): Coherent Measures of Risk, *Mathematical Finance* 9 no. 3, 203–228.
3. Deák I.(2001): Successive regression approximations for solving equations. *Pure Mathematics and Applications* 12, 25–50.
4. Deák I.(2002): Computing two-stage stochastic programming problems by successive regression approximations. In *Stochastic optimization techniques*, vol. 513 of Lecture Notes in Econom. and Math. Systems, Springer, Berlin, 91-102.
5. Deák I. (2004): Solving stochastic programming problems by successive regression approximations – numerical results. In *Dynamic stochastic optimization*, vol. 532 of Lecture Notes in Econom. and Math. Systems, Springer, Berlin, 209-224.
6. Deák I.(2006): Two-stage stochastic problems with correlated normal variables: computational experiences, *Annals of Operations Research*, 142, 79–97.
7. Fábián Cs., Veszprémi A.(2007): Algorithms for handling CVaR-constraints in dynamic stochastic programming models with applications to finance. *The Journal of Risk* 10, 111–131.
8. A. Künzi-Bay, J. Mayer (2006): Computational aspect of minimizing conditional value-at-risk. *Computational Management Science* 3, 3–27.
9. W.-K. Mak, D. Morton, R. Wood (1999): Monte Carlo bounding techniques for determining solution quality in stochastic programs, *Operations Research Letters*, Volume 24, Number 1, 47–56.
10. G. Pflug (2000): Some remarks on the Value-at-Risk and the Conditional Value-at-Risk. In *Probabilistic constrained optimization* (ed. Uryasev), Kluwer, Dordrecht, 272–281.

11. Prékopa A.(1973): Contributions to the theory of stochastic programming. *Mathematical Programming*, Vol. 4, No. 1, 202-221.
12. T. Rockafellar, S. Uryasev (2000): Optimization of Conditional Value-At-Risk. *The Journal of Risk*, Vol. 2, No. 3, 21–41.
13. T. Rockafellar, S. Uryasev (2002): Conditional Value-at-Risk for general loss distributions. *Journal of Banking & Finance* 26, 1443–71.

CVAR MINIMIZATION BY THE SRA ALGORITHM

The risk measure *CVaR* is becoming more and more popular in recent years. In this paper we use *CVaR* for portfolio optimization. We formulate the problem as a two-stage stochastic programming model. We apply the SRA algorithm, which is a recently developed heuristic algorithm, to minimizing *CVaR*.